GPGPU (General Purpose Graphical Processing Unit) を用いたソフトウエア高速化の試み

Acceleration of Software Using GPGPU (General Purpose Graphical Processing Unit)

小島　航*　　角谷　浩享
Wataru OJIMA　　Hiroyuki KADOTANI

(平成20年10月 8 日受理)

板状Siに打ち込まれた高速中性子の挙動を模擬するモンテ・カルロプログラム[1]をGPGPUによる並列処理で高速化を試みた。ホストPC上のCPUを用いる場合との比較で、同等以上の処理速度を確認できた。

1．はじめに

Fortranで書かれた科学計算向けソフトウエアは、膨大な計算時間が必要なものが多いが、実行時間の大半がプログラム中の一部分に集中していることがある。これらのプログラムは、時間のかかる一部分を改良することで計算時間の短縮ができる。改良の方法には、並列化やベクトル化などがあり、大型計算機や、PCクラスターなどを利用することで処理速度を上げることができる。また、FPGAや専用アクセラレータチップを応用する方法も注目されている。一方でPCに搭載されているグラフィックス処理チップGPUを汎用目的に応用し、処理速度を高めるGPGPUというものが出てきた。本研究ではFortranプログラムをLinux上でCUDAを用いてGPGPU向けに改良したソフトウエアを作成し、その特性を調べた。

2．GPGPUとは

GPGPU (General Purpose Graphical Processing Unit) は、GPU (Graphics Processing Unit) の持つ、高い処理能力を汎用目的に応用するものである。GPUとはグラフィックス専用の高速演算チップである。GPUは３Dグラフィックス処理の手法の進歩に伴い、プログラム可能な演算回路を持つようになった。また、IEEE754仕様の浮動小数点の実装など、汎用CPUに近い機能を実装するようにもなった。汎用CPUと同様に、高い処理能力を要求されつづけているため、性能向上のペースも非常に早い。CPUやFPGA、メモリーチップなどと同様に、最新の半導体プロセス技術を応用した製品の１つでもある。初期のGPGPUは、GPUのもつプログラム可能な演算回路プログラマブルシェーダーを活用するために、３Dグラフィックスプログラムを応用して作られている。プログラマブルシェーダーを応用したプログラムは、難易度の高いものであり、３Dグラフィックスプログラム

＊　本学国際情報学部2002年度卒業生

に関する、専門的な知識が必要であった。また、GPUに汎用的な処理をさせる上での制限も多かった。CUDAとは、NVIDIA社が自社GPU向けに提供しているGPGPU開発環境である。CUDAは、C言語をベースにした開発環境であり、標準的なC言語にGPGPU向け拡張を実装している。従来の3Dグラフィックスプログラムベースのものとは全く違い、C言語や、並列処理に関する知識があればGPGPU向けソフトウエアを作成できる。

3．実験に使用した機材について

実験には下記の仕様のPCとビデオボードを使用した。OSにはFedora 8 Linux AMD64を使用した。ホストPCのCPUとGPUはともに65nmの微細加工プロセスを使用した同世代の半導体チップでもある。コンパイラはCUDA用nvcc、Cコンパイラ GCC4.2、Fortranコンパイラには Gfortran4.2を使用した。

ホスト：PC：Intel Core 2 Duo（65nm）1.8Ghz、IntelX38チップセット、DDRII800メモリー4 GB

ビデオボード：Geforce8800GTS（65nm）512MB（256bit）

4．1　実験内容について

今回は、板状Siに打ち込まれた高速中性子の挙動を模擬するモンテ・カルロプログラム[1]をGPU上で並列処理することで、高速化を試みる。このプログラムの特徴は、計算処理部分に相互依存する部分が少ないことにある。GPU上の汎用プロセッサは、独立して計算処理を実行することができる。

4．2　プログラムの構造について

プログラムの構造は、Fortranプログラムが本体であり、ここからCUDA向けのC言語プログラム部分をサブルーチンとして呼び出している。CUDAは現状ではC/C++言語のみをサポートしているため、CUDA上に実装する部分をC言語に変換する必要がある。Linux向けCUDAではGCCをベースにしているため、プログラム自体は、他の言語をベースにしたものでも利用ができる。本文では科学計算向け言語Fortranで作成されているプログラムを、計算部分のみC言語に書き換え、これをCUDA向けに改造した。

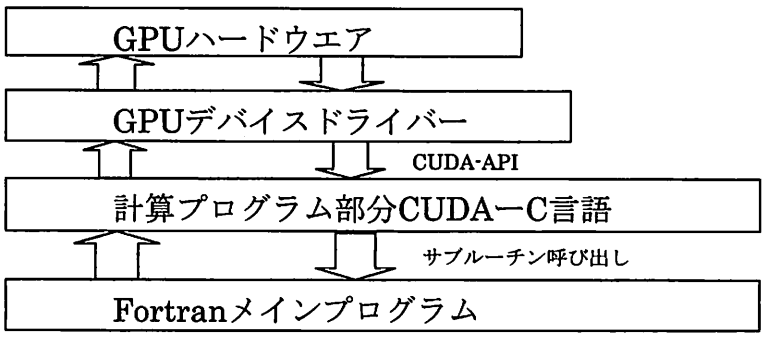


図1　プログラムの構造

ハードウエアからの視点では、CUDAで利用するメモリー空間はCPUのものとは独立したビデオカード上のフレームバッファメモリーに存在し、ホストPC上のCPUとビデオカード上GPU間のデータの交換は、CUDAで用意されたAPIを利用する。計算開始時には、ホストPCからフレームバッファメモリーにデータ転送を行い、計算結果をフレームバッファメモリーからホストPCに転送し、処理結果を得る。

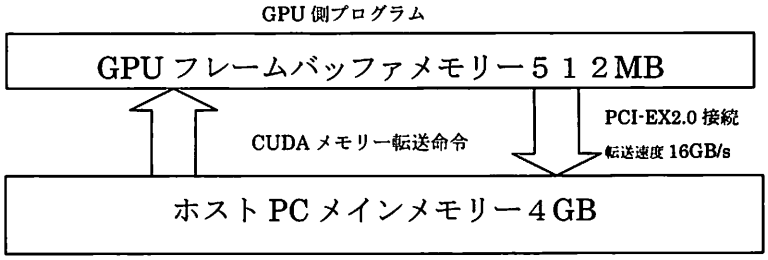
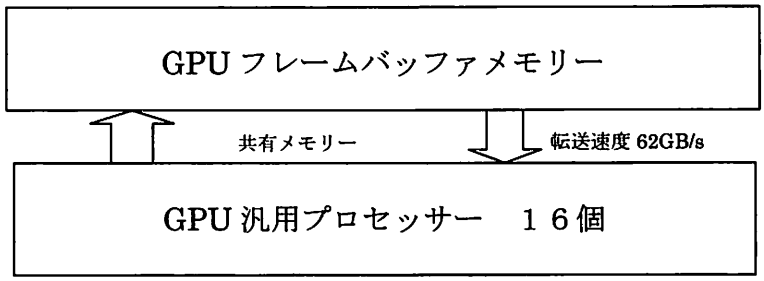


図２　ホストPCとGPUのメモリー空間の関係

CUDA環境からの視点では、今回使用したGeforce8800GTSは、16個の汎用プロセッサで構成された並列処理プロセッサのように見える。各プロセッサ上では同じプログラムが同時に実行されており、フレームバッファ上のメモリーは、各プロセッサ間で共有した空間でアクセスできる。GPU上のプログラムは自身のプロセッサ番号を識別できるため、これを元に分散処理をしている。



汎用プロセッサ１つは８つのプログラマブルシェーダーで構成されているので
１２８個のプログラマブルシェーダーが存在している

図３　CUDA上でのGPU汎用プロセッサーとメモリー

５．計算結果

ホスト側のCPU Intel Core 2 Duo 1.8Ghzと処理時間を比較した。800万回ループ計算をGPU上で並列実行した場合、約６秒、ホスト側CPUで逐次実行した場合約11秒であった。また、ホストCPUのクロックを調整した場合、GPUでの実行時間はホストCPU側の処理速度には関係していないことも確認した。計算回数が160万回まではCPUの方が処理時間が短く、それ以上の回数ではGPUでの処理時間が短い結果になった。これは、ホストCPUとGPU間の通信や、GPU上でプログラム実行までにかかる時間により、効率が落ちた為と考えられる。

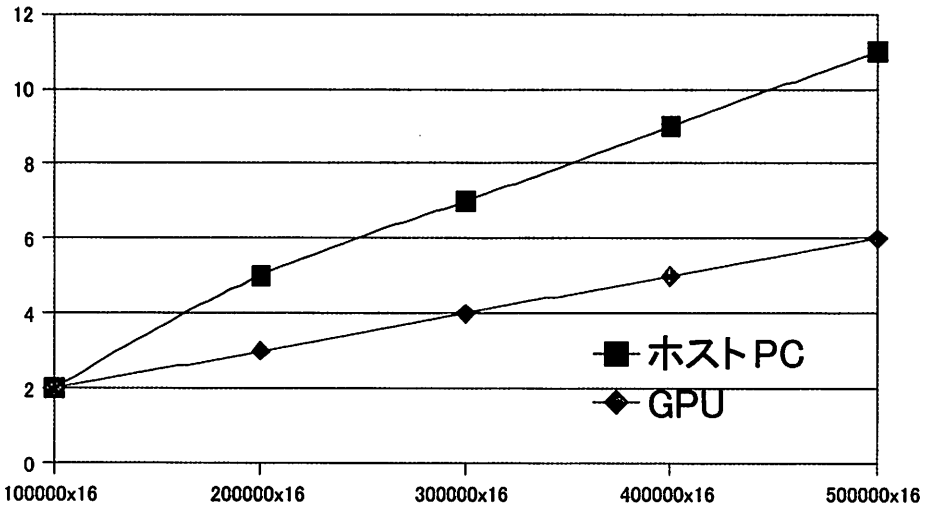


図4　ホストPCとGPUの処理時間の比較（縦軸は秒、横軸は計算回数）

6．まとめ

板状Siに打ち込まれた高速中性子の挙動を模擬するモンテ・カルロプログラムにおいて、GPGPUと汎用CPUを比較し、同等以上の処理速度を確認できた。今回のプログラムは1CPUと1つのGPUを搭載したビデオカードとの比較であったが、CPU側ではマルチプロセッサー構成や、クラスタリングなどにより、高速化が可能である。一方、GPU側もより高速なGPUを搭載したビデオカードが存在し、1台のホストPCに複数枚のビデオカードを搭載することで高速化ができる。また、CUDA専用に調整された演算用カードや、システム製品も存在する。MPIなどのクラスタリング環境との共存もできるため、GPUを複数搭載したホストPCをクラスタリングし、規模の大きなシステムも作成可能である。

今回は、対象としたプログラムをCUDAを用いてGPGPUで実行した。元のプログラムはFortranで作られていたため、C言語に変換する手間はかかったが、C言語からCUDA向けへの改造は比較的容易であった。初歩的なものではあるが、筆者は、マルチプロセッサ[2]、MPIクラスタリング[3]、FPGAを用いた並列化[4]の経験がある。これらの経験からみて、CUDAを用いたGPGPU向け処理への改造は、マルチプロセッサよりは難易度が高いが、MPIクラスタリングよりも容易であった。FPGAと比較すれば非常に容易な作業であった。マルチプロセッサやMPIクラスタリングでは、CPUやソフトウエアは同じものを利用するため、互換性の問題は起こらない。しかしGPUでは実装されている機能がCPUとは完全互換ではない。今回は互換性が問題にならないように、GPUの機能と同等のものをCPU上のソフトウエアでも利用した。互換性の問題とは、1つは倍精度型浮動小数点が未実装であり、これは今回使用したGPUの次世代品では実装されている。もう1つの問題は、ソフトウエア上での互換性である。現状のCUDAではC/C++言語とCUDAで用意されている機能のみGPU上で実行できる。単純な数学関数はC言語のMath.h程度には用意されているので、今回のプログラムでは問題にならなかったが、実装されていない機能は、作業者自身でGPU上にC言語で実装するか、ホストPC上のCPU処理で代行す

る必要がある。

今後の課題としては

・複数枚のビデオカードを使用したマルチGPU環境での並列化による高速化

・ホストCPUと連動した形でのGPUの利用

・上記を応用したうえで、ホストPCをMPI等でクラスタリング構成をし、より高度な並列化による高速化

・実用的なソフトウエアからのGPGPU応用ソフトウエアの利用方法を探す

などが考えられる。

参考文献

(1) Norman Schaeffer, Edit., "Reactor Shielding for Nuclear Engineer", p.679, U.S. Atomic Energy Commission, 1973, より変更して作成

(2) 小島 航, 角谷 浩享, "多倍長整数演算の並列化", 静岡産業大学国際情報学部研究紀要 第5号 p.31, 2003

(3) 小島 航, 角谷 浩享, "並列計算機の試作とその性能評価", 静岡産業大学国際情報学部紀要第6号 p.45, 2004

(4) 小島 航, 角谷 浩享, "3x+1問題に対するFPGAの利用", 静岡産業大学情報学部紀要 第9号 p.13, 2007

(5) CUDAに関する一般的な参考文献として、以下のものを参照した。
NVIDIA_CUDA_Programming_Guide_2.0 NVIDIA
CudaReferenceManual_2.0 NVIDIA